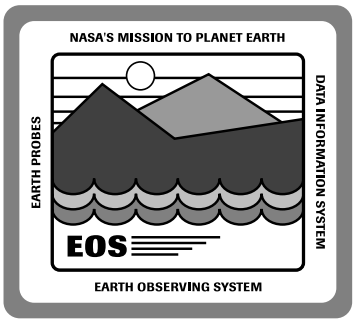
430-TP-006-001



PDPS Prototyping at the ECS Science and Technology Laboratory: Report #5

Technical Paper

April 1995

Prepared Under Contract NAS5-60000

RESPONSIBLE ENGINEER

Narayan S. Prasad 4/10/95

Narayan S. Prasad, PDPS Scientist/Engineer Date
EOSDIS Core System Project

SUBMITTED BY

Parag Ambardekar 4/10/95

Parag Ambardekar, TRMM Development Manager Date
EOSDIS Core System Project

Hughes Applied Information Systems
Landover, Maryland

This page intentionally left blank

Abstract

Using heritage ECS science software, we assess the various processing alternatives currently available. In this study, we focus on the applicability and suitability of the newer Massively Parallel Processors (MPPs) like the Cray T3D and IBM SP2 for ECS science algorithms. The objective of this exercise is to demonstrate by prototyping ways to develop and migrate parallel applications from Symmetric Multiprocessors (SMPs) and workstation clusters to MPPs. This study is based on heritage SeaWinds Level 2 ECS science software from the Jet Propulsion Laboratory (JPL). The SMP class of machines currently recommended for the Data Processing Subsystem at the Distributed Active Archive Centers (DAACs) for Ir1 provide the capability to do traditional sequential processing and also multiprocessing of science data. The SMPs are projected to approach the capabilities of MPPs in the near future. Nevertheless, it is important to assess the status of MPPs that are currently available. With substantial increase in processing requirements called for Releases B and beyond, it is essential to evaluate the newer MPPs and study ways to migrate applications developed on SMPs, workstation clusters to MPPs. Some initial deliveries of science software may subsequently evolve into "Tall Poles" for Release B. The traditional MPPs were difficult to program and made code less portable. Over the years, the unavailability of parallelization tools also contributed to the difficulty in parallel software development on MPPs. Therefore, as part of this prototyping, we study the newer MPPs like the Cray T3D (at JPL) and IBM SP2 (at NASA/Ames). The actual floating point performance as a function of number of processors is calculated for each hardware platform on the basis of SeaWinds execution in both sequential and multi-processing modes. Performance characteristics derived in this study will be utilized for ECS systems performance modeling and Data Processing System hardware sizing. Also, the maturity of parallelization tools available on MPPs are assessed. This informal write-up documents our experiences and findings from prototyping.

Contents

1. Introduction

1.1 Purpose 1-1

1.2 Background 1-1

1.3 Review and Approval 1-1

2. Characterization of SeaWinds Algorithm 0-1

2.1 Background 2-1

2.2 Data sets 2-1

2.2.1 Input data sets 2-1

2.2.2 Output data sets 2-1

2.3 Processing 2-1

2.3.1 Initialization 2-1

2.3.2 Computation of wind parameters 2-2

2.3.3 Output product generation 2-2

3. Analysis

3.1 Introduction 3-1

3.2 Develop Parallelization Strategy 3-1

3.3 Performance Analysis 3-1

3.3.1 SGI Challenge XL 3-1

3.3.1.1 Architecture 3-1

3.3.1.2 Parallelization Environment/Tool 3-1

3.3.1.3 Timing 3-2

3.3.1.4 Speedup 3-2

3.3.1.5 Floating Point Performance 3-4

3.3.2 Workstation Cluster 3-4

3.3.2.1 Architecture 3-4

3.3.2.2 Parallelization Environment/Tool 3-5

3.3.2.3 Parallel Performance Analysis 3-5

3.3.2.4 Automatic Parallelization 3-6
3.3.2.5 Interactive Fortran Parallelization 3-6
3.3.2.6 Timing 3-6
3.3.2.7 Speedup 3-7
3.3.2.8 Floating Point Performance 3-8
3.3.3 Performance Analysis on IBM SP2 3-9
3.3.3.1 Architecture 3-9
3.3.3.2 Parallelization Environment/Tool 3-10
3.3.3.3 Timing 3-10
3.3.3.3.1I/O On Multiple Nodes 3-10
3.3.3.3.2I/O On One Node 3-10
3.3.3.4 Speedup 3-11
3.3.3.5 Floating Point Performance 3-12
3.3.4 Cray T3D 3-12
3.3.4.1 Architecture 3-12
3.3.4.2 Parallelization Environment/Tool 3-13
3.3.4.3 Timing 3-13
3.3.4.4 Speedup 3-14
3.3.4.5 Floating Point Performance 3-15
3.3.2 Performance Comparison of Various Hardware Platforms 3-17

4. Caveat

5. Purdue Benchmarks

5.1 Background 5-1
5.2 Purpose 5-1
5.3 SGI Challenge 5-1
5.4 Workstation Cluster 5-1
5.5 Cray T3D 5-3
5.6 IBM SP2 5-3

6. Summary and Conclusions

7. References

8. Acknowledgments

Abbreviations and Acronyms

1. Introduction

1.1 Purpose

This report documents the recent activities of the ECS Science and Technology Laboratory (STL) prototype for science software execution. It is the one of the series of reports published for the informal dissemination of technical information to ESDIS, Instrument Teams (ITs) and science software developers. The purpose of these activities is to assess various processing alternatives, evaluate and validate hardware architecture for the Data Processing Subsystem.

1.2 Background

As part of the prototyping activities at the ECS STL, we acquired several representative and heritage ECS science software to study various processing alternatives (e.g. OSF/Distributed Computing Environment (DCE) [1] [2], symmetric multiprocessing [3], distributed memory processing and massively parallel processing) available for processing ECS data. This activity demonstrates by prototyping ways to develop and migrate parallel applications from Symmetric Multiprocessors (SMPs) and workstation clusters to Massively Parallel Processors (MPPs).

1.3 Review and Approval

This document is an informal report of Planning and Data Processing System (PDPS) prototyping activities. It does not require review and approval by the government. Questions regarding technical information contained within this paper should be addressed to the following ECS and or GSFC contacts:

- ECS Contact

  Narayan Prasad, PDPS Scientist/Engineer

  (301)-925-0467

  nprasad@eos.hitc.com

- GSFC Contact

  Tonjua Hines, Hardware Manager

  (301)-286-8807

  thines@ulabsgi.gsfc.nasa.gov

2. Characterization of SeaWinds Algorithm

2.1 Background

The NASA Scatterometer II (NSCAT II) scheduled to be launched aboard the Advanced Earth Observing System (ADEOS) II spacecraft will acquire accurate, high resolution, continuous all-weather measurements of near-surface vector winds over the ice-free global oceans crucial for studies of tropospheric dynamics and air-sea momentum fluxes. The NSCAT II data products will consist of global multi-azimuth normalized radar cross section measurements; 25-km^2 resolution ocean vector winds in each of the swaths. These multiple measurements from multiple angles will be processed by SeaWinds Level 2 algorithm to invert the backscattered signal to derive wind speed and direction.

The SeaWinds package used for Planning and Data Processing System Software Execution prototyping consists of a program that models the processing of radar backscatter measurements to wind vectors. The retrieval of wind vectors is the most CPU-intensive portion of the processing. The program encompasses most of the algorithm functions assigned to Level 2 wind vector processing. The normalized radar cross sections for wind retrieval are grouped into wind vector cells (20-80 per cell). The Advanced Microwave Scanning Radiometer (AMSR) data correction factors are applied, wind vectors retrieved and an unique solution selected by removing any ambiguities.

2.2 Data sets

2.2.1 Input data sets

The input data sets for SeaWinds Level 2 processing are:

- Simulated measurements of normalized radar cross sections for one-third of a revolution (using a SeaWinds engineering performance model "flown" over a numerical weather model wind field) which have been time-tagged and earth-located (9 MB)
- Digital Weather Model (DWM) Data (2 MB)
- Tabulated radiation scattering model function (0.4 MB)
- Parameter control files (< 1 MB)

2.2.2 Output data sets

- Level 2 near-surface ocean wind vectors per swath over the ocean (5 MB)

2.3 Processing

2.3.1 Initialization

The initialization routines read DWM data and backscattering model tabulated functions, sets processing parameters, initializes variables and arrays, and selects accuracy and mode of data processing.

2.3.2 Computation of wind parameters

The processing for wind parameters is performed using a "rolling buffer" to collect all data relevant to a particular geolocated cell from chronologically organized input data files. The following highlight the computation:

- Group normalized radar cross section for wind retrieval into wind vector cells
- Apply AMSR correction factor
- Retrieve wind vectors
- Select a unique solution by removing ambiguities

2.3.3 Output product generation

The output data are generated in small bursts with each burst equal to the size of the rolling buffer (area equal to the field-of-view of the instrument).

3. Analysis

3.1 Introduction

The SeaWinds Level 2 wind retrieval (henceforth called SeaWinds) takes about half the total end-to-end Level 2 processing time. Therefore, we target this portion of the algorithm to study suitability to parallel processing for increasing throughput. After source code analysis, we analyzed performance by first running in sequential mode on an SGI Challenge XL.

3.2 Develop Parallelization Strategy

The SeaWinds Level 2 wind retrieval processing is performed by first organizing data for one-third of a revolution into smaller cells. Each cell is processed independently of other cells. A closer analysis of SeaWinds indicates that such a cell-based approach allows for processing to be performed in parallel using a multiprocessor machine. In other words, the algorithm lends itself to medium- and fine-grained parallelism. In contrast, in our earlier prototype [3], the Pathfinder SSM/I algorithm was parallelized in a coarse-grained (at the orbit level) fashion. The steps to a parallel SeaWinds program is similar to the steps outlined for the Pathfinder SSM/I Precipitation Rate algorithm [3]. The code has 3.5 Giga Floating Point Operations (integer and floating point operations are grouped together). Roughly 3.2 Giga Floating Point Operations (92% of the Level 2 software) can be run in parallel.

3.3 Performance Analysis

3.3.1 SGI Challenge XL

3.3.1.1 Architecture

The Challenge XL is the high end member of the Challenge line. The XL server supports coherent shared memory and symmetric multiprocessing based on 100 MHz and 150 MHz MIPS RISC R4400MC 64-bit microprocessors. The server can be configured with 2 to 36 CPUs, 64 MB to 16 GB of main memory, 2 GB to 3.4 TB of disk capacity, one to four 320 MB/sec I/O channels, and five to 25 industry-standard VME64 bus slots. The STL has an 8-processor Challenge XL loaned by Silicon Graphics, Inc. for evaluation purposes.

3.3.1.2 Parallelization Environment/Tool

The Power Fortran Accelerator (PFA) [4] is a Fortran 77 source-to-source preprocessor that enables existing Fortran 77 programs to run efficiently on SGI POWER Series™ multiprocessor systems. The PFA analyzes a program and identifies loops that do not contain data dependencies. Such loops are safe to execute in parallel. The PFA automatically inserts special compiler directives in a modified copy of the original source code. The PFA also produces a number of files containing code and other information to run the program concurrently on multiple processors. Because the directives inserted by the PFA look like standard Fortran 77 comment statements, it does not affect the portability of the code to non SGI systems. The listing file optionally generated by the PFA can be used to identify potential data dependencies that prevented the PFA from running a loop in parallel. The object files prepared

by the PFA are fully compatible with object files prepared for a serial compiler. They can be freely combined for execution.

The PFA analyzes a program for data dependence. During this analysis, the PFA looks for Fortran 77 DO loops in which each iteration of the loop is independent of all other iterations. If each iteration of the loop is self-contained, the system can execute the iterations in any order (or even simultaneously on separate processors) and produce the same result after running all iterations.

When the PFA finds a loop with data independence, it knows if the loop can be safely run in parallel. When the PFA finds a loop that contains iterations that are dependent upon other iterations, it cannot safely run the loop in parallel, but the PFA can tell what is causing the problem. If the PFA cannot run the loop in parallel, the listing file will explain where it encountered problems. The programmer can make changes to the program by eliminating dependencies or restructuring sections of the code.

3.3.1.3 Timing

Figure 3.3-1 illustrates the execution time for SeaWinds compiled using the PFA and executed on 1 to 8 processors. All input data were read from a host attached disk by only one processor (processor 0). The geolocation of all input data were performed serially because there is no inherent parallelism in this portion of the algorithm. However, the wind retrieval and ambiguity removal were performed in parallel. The overhead in execution time is system related. This includes any time to set up resources before program execution. Because communication among processors is implicit in a shared memory architecture like the SMP, there is no explicit data distribution from processor 0 to all other processors. This is reflected by the dark shaded area in Figure 3.3-1. As the number of processors increased, the time taken to execute the serial code and I/O is constant. This is in sharp contrast to distributed memory machines (discussed later) where data are read by the first processor and subsequently distributed to all other processors to begin parallel execution. The parallel sections of the program, however, shows nice scaling and directly contributes to the overall improvement in performance.

3.3.1.4 Speedup

The speedup is defined as the ratio of the total execution time on *n* processors to that on a single processor. Figure 3.3-2 shows speedup as a function of number of processors. The ideal speedup is shown for reference as a solid line. The solid line with squares and dashed line in Figure 3.3-2 illustrates speedup for the parallel sections and the overall program, respectively. As expected, the parallel sections exhibit almost a linear speedup. The overall speedup is also impressive (4.5 on 8 processors).

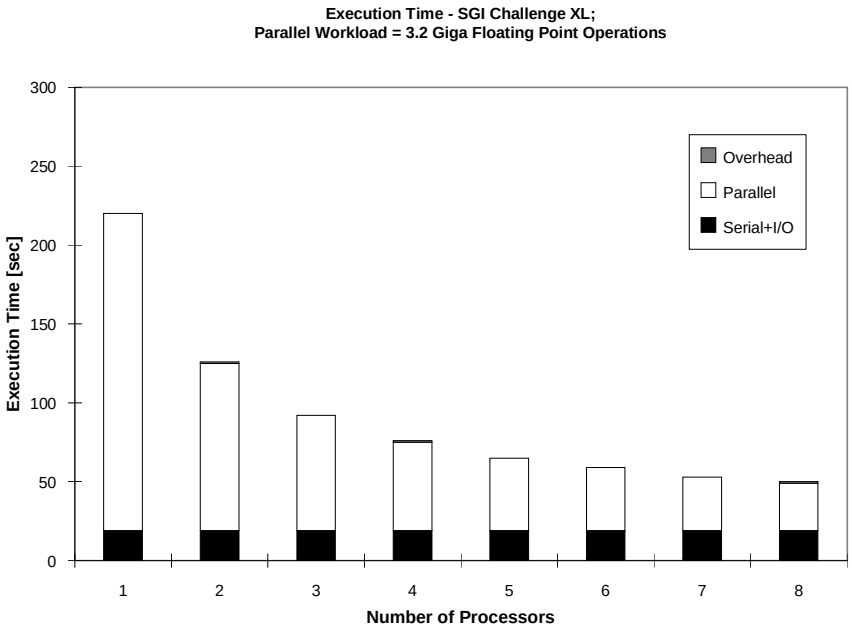


Figure 3.3-1. SeaWinds Execution Time on SGI Challenge XL

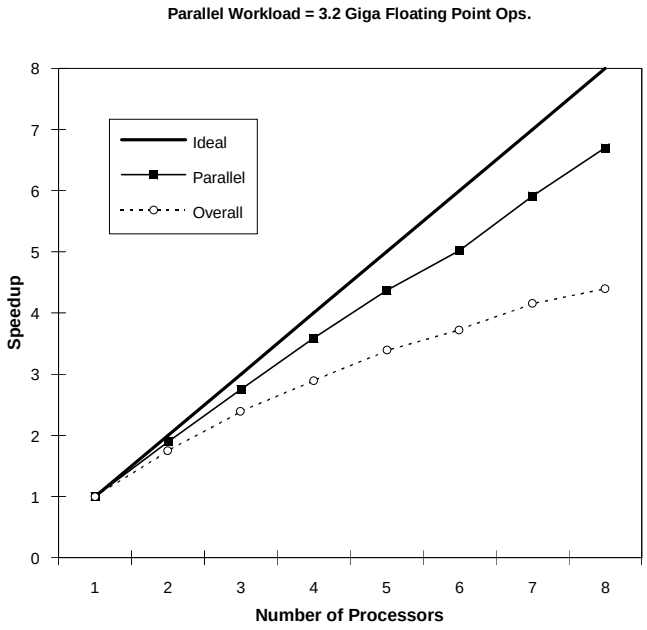


Figure 3.3-2. SeaWinds Speedup on SGI Challenge XL

3.3.1.5 Floating Point Performance

For the purposes of ECS Systems Performance Modeling and hardware sizing, it is important to measure floating point performance of a hardware based on real science software. Millions of Floating Point Operations (MFLOPS) was selected as a measure because the current version of the ECS Systems Performance Model accepts performance figures in vendor rated peak MFLOPS. The vendor rated MFLOPS can then be adjusted by including a processing efficiency factor for both sequential and multiprocessing alternatives. The performance estimation is based on the parallel sections of the program with 3.2 Giga Floating Point Operations. The vendor rated peak theoretical floating point performance for SGI Challenge XL is 75 MFLOPS per processor. Figure 3.3-3 illustrates total floating point performance (left scale) and average floating point performance per processor (right scale). On the basis of single and multiprocessor runs of SeaWinds, the total MFLOPS ranged from 15 for a single processor to just over 100 with all 8 processors. This averages to about 14 MFLOPs per processor, about ~20% processor efficiency. Unlike MPPs (discussed later), the average floating point performance per processor does not drop with increase in number of processors.

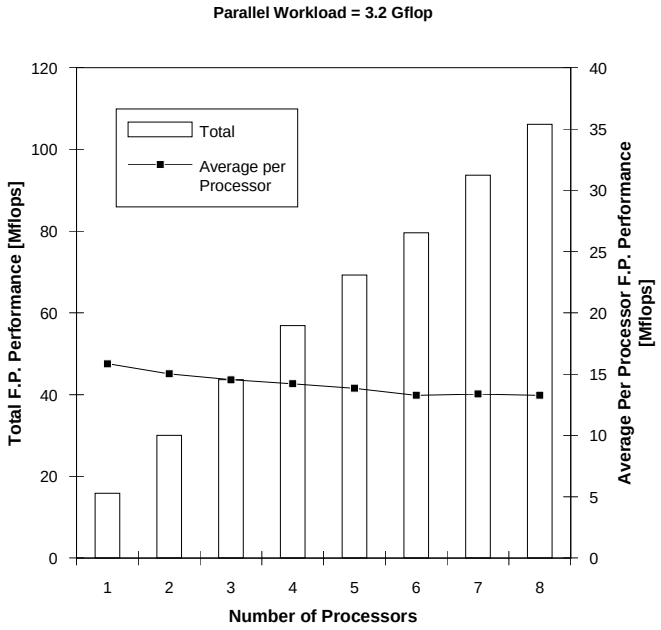


Figure 3.3-3. Floating Point Performance of SGI Challenge XL for SeaWinds

3.3.2 Workstation Cluster

3.3.2.1 Architecture

The ECS STL distributed/parallel testbed consists of a cluster of workstations with characteristics as shown in Table 3.3-1. The workstations are connected together by a Fiber Distributed Data Interface (FDDI) with a peak transfer rate of 100 Mb/s.

3.3.2.2 Parallelization Environment/Tool

The parallelization tools used for creating parallel and distributed applications are xHPF and FORGE 90 Distributed Memory Parallelizer (DMP) from Applied Parallel Research (APR). FORGE 90 is an interactive GUI-based parallelizer, while xHPF is a batch parallelizer. Their features include:

- Automatically converting a Fortran 77 program into a parallel Single Program, Multiple Data (SPMD) program to run on multiple processors.
- Parallelization of DO loops
- Interactive review of parallelization strategies using FORGE interactive DMP
- Parallel runtime performance analysis for locating interprocessor communication bottlenecks and load balancing problems
- Support of several message passing libraries including Parallel Virtual Machine (PVM), Linda, and Express

Table 3.3-1. Workstation Cluster Characteristics

Characteristics	DEC 3000/300	HP715/50	HP735	SGI R4000	Sun Sparc 10	IBM RS6000/34H
No. of workstations	1	1	2	1	3	1
Operating system	OSF/1	HP-UX	HP-UX	IRIX	Solaris	AIX
CPU architecture	Alpha AXP	PA-RISC with integrated floating point co-processor	PA-RISC with integrated floating point co-processor	MIPS R4000	SuperSparc	RISC
No. of processors	1	1	1 (each)	1	(1 each)	1
Clock speed	150 MHZ	50 MHZ	99 MHZ	100 MHZ	40 MHZ	50 MHZ
Cache	512 KB	64 KB	256 KB	16 KB	36 KB	32 KB
Peak performance	91 SPECfp 92 66 SPECint 92 24 MFLOPS	72 SPECfp 92 36 SPECint 92 13 MFLOPS 69 SPECmarks 62 MIPS	150 SPECfp 92 80 SPECint 92 40 MFLOPS 147 SPECmarks 124 MIPS	61 SPECfp 92 57 SPECint 92 16 MFLOPS	60.2 SPECfp 92 50.2 SPECint 92 22.9 MFLOPS	64.5 SPECfp 92 29.7 SPECint 92 15 MFLOPS

3.3.2.3 Parallel Performance Analysis

Both FORGE 90 and xHPF allow for the instrumentation of parallelized programs they generate to produce a timing report. When run on a target platform, they profile the program's parallel performance and identify data communication as well as routine and loop timings. The parallelized program can thus be fine tuned by restructuring the code or inserting directives to alter data partitioning or loop distribution decisions.

3.3.2.4 Automatic Parallelization

This is the first step in parallelizing a Fortran program by using DMP's automatic array distribution capability to derive an initial parallelization strategy. The FORGE 90 DMP's Data Decomposition facility further refined the parallelization by interactively specifying decomposition and selecting arrays for partitioning while viewing the implications of these decisions. The Data Decomposer implements either BLOCK or CYCLIC distributions along multiple dimensions, with either FULL or SHRUNK memory allocation. With *full* allocation, an array is allocated in its original size on each processor. With *shrunk* allocation, each processor is allocated only enough memory for an array to hold the elements that it owns. The DMP's Loop Spreader allows interactive or automatic selection of loops. Under automatic selection, DMP uses actual runtime execution statistics and the appearance of distributed arrays to determine the best loops to parallelize, obtaining higher parallelization granularity and reduced communication costs.

3.3.2.5 Interactive Fortran Parallelization

The DMP was used to spread loops and distribute arrays across clusters of networked workstations *interactively*. The parallelized program is fully scalable, with calls to APR's parallel run-time library, interfacing PVM or Message Passing Interface (MPI). With the SPMD parallelization strategy, the same program runs on each compute node while selected DO loops are rewritten to automatically distribute their iterations across the clustered nodes.

3.3.2.6 Timing

The execution times for a cluster of SGI Indigo, two HP735s and HP715 in that order is illustrated in Figure 3.3-4. PVM was used for message passing among processors. Data were read from a host attached disk. In a heterogeneous cluster comprising of machines of different makes and speed, ordering of nodes can impact overall performance. This is because the assignment of work (load balancing) is determined at run time. The slower processor can be assigned work that is more intensive and the faster processor can be assigned a smaller workload. Load balancing is not performed automatically by the parallelization tools. However, the programmer has the choice of decomposing a problem (BLOCK or CYCLIC decomposition of arrays) using the parallelization tools to aid a more even distribution of load. An optimal configuration is determined iteratively. As shown in Figure 3.3-4, the optimal configuration that we obtained achieved speedup comparable to the SGI Challenge XL. However, unlike the SGI Challenge XL where only one processor performed all read/write operations, we allowed each workstation on the cluster to perform I/O operations in parallel. Each processor ultimately owned the data it processed. Because data were read concurrently by each node, no data were distributed to the other processors across the network. Concurrent or parallel I/O can reduce data distribution overhead substantially.

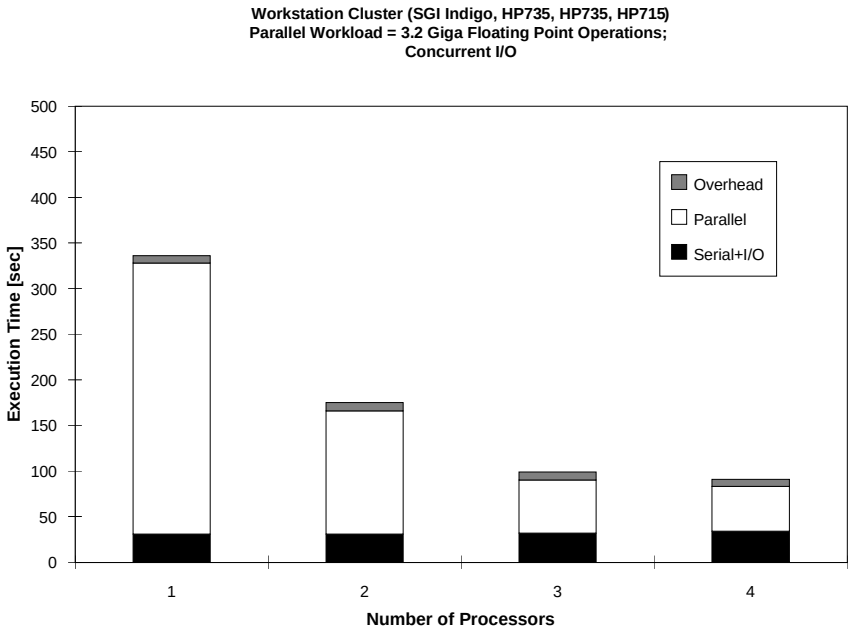
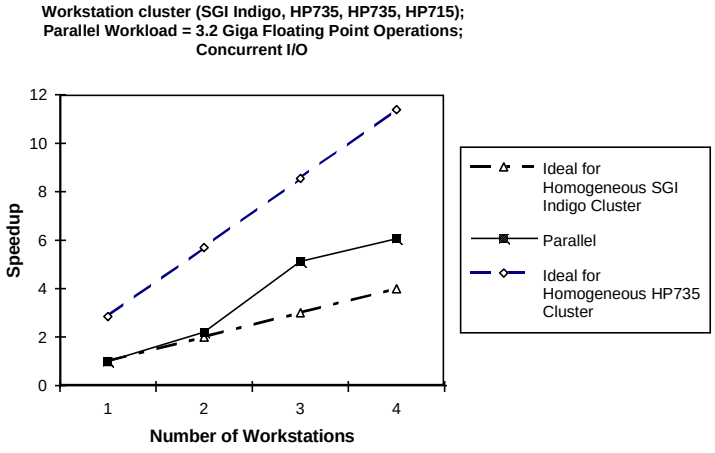


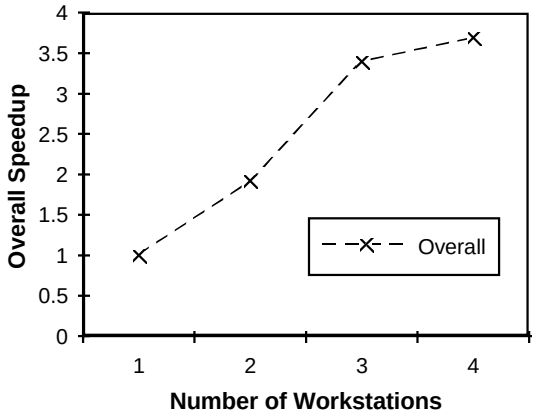
Figure 3.3-4. SeaWinds Execution Time on Workstation Cluster

3.3.2.7 Speedup

The speedup is defined as the ratio of the time taken for the program to execute on *n* processors to that on a single processor. Figure 3.3-5 upper and lower panels illustrate the speedup of SeaWinds for the parallel sections and the overall program, respectively. If a cluster is homogeneous (i.e. all machines in the cluster are identical), then the ideal speedup is a simple straight line relationship. If a cluster is heterogeneous (i.e. it is made up of processors that are



**** Speedup is dependent on ordering of nodes**



*** If processors of different clockspeeds are used, we would have a lower bound and an upper bound that encompass the actual speedups. The HP735 was ~3 times faster than SGI Indigo. The slowest machine (SGI) in the cluster determines the lower bound, while the fastest machine (HP735) determines the upper bound.**

Figure 3.3-5. SeaWinds Speedup on Workstation Cluster

not identical), then speedup is dictated by the slowest and the fastest machines on the cluster. On the basis of performance comparison using several science software (Pathfinder AVHRR/Land, Pathfinder SSM/I Precipitation Rate and SeaWinds), we determined that the HP735 is approximately 3 times faster than the SGI Indigo. Therefore, the ideal speedup for a hypothetical cluster of HP735 workstations is represented as the upper bound, and the ideal speedup for a hypothetical cluster of SGI Indigo workstations is represented as the lower bound (dashed lines in the upper panel of Figure 3.3-5). The HP715 and SGI Indigo workstations are comparable in their speed. The speedup was normalized to the SGI Indigo. When SeaWinds executed on two processors (SGI Indigo and HP735), the workload was approximately split in half. This explains a speedup slightly greater than 2. The slower processor limits any dramatic increase in the speedup. With an additional HP735, work was divided three-way with approximately two-thirds of the total work performed by the faster HP735 workstations. The slope of the speedup curve is now almost parallel to the slope of the ideal curve representing a cluster of HP735 workstations. Addition of a slower HP715 workstation forced the speedup to approach the lower bound. Note that data were read concurrently on each node. Therefore, each node owned the data it processed. Figure 3.3-5 (lower panel) illustrates the overall speedup.

3.3.2.8 Floating Point Performance

Figure 3.3-6 illustrates the cumulative floating point performance (left scale) and the average floating point performance per processor (right scale). The rated peak MFLOPs of each processor is also represented. The floating point performance was calculated for the parallel sections of the code with 3.2 Giga Floating Point Operations workload. This translates to an average of 34% efficiency for the entire cluster.

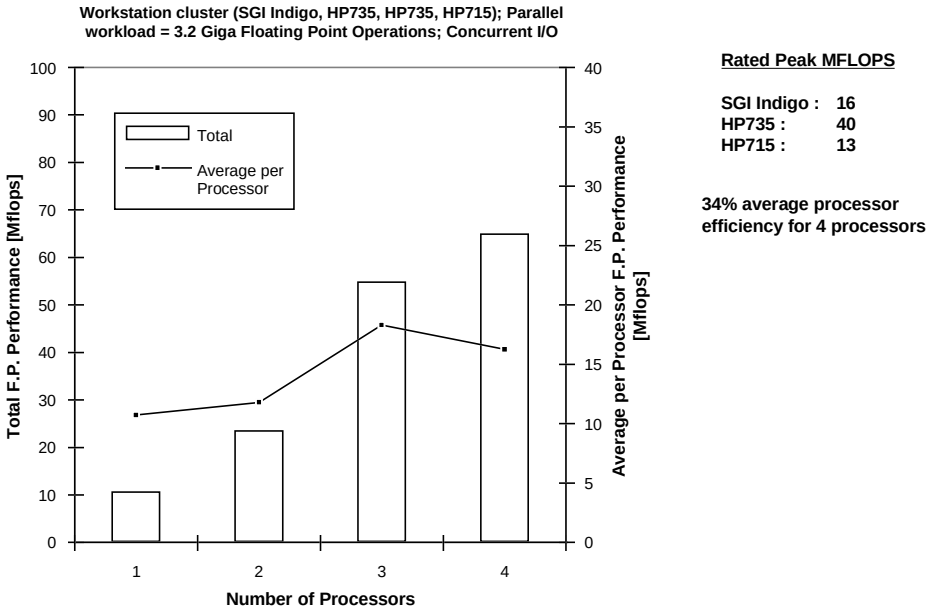


Figure 3.3-6. Floating Point Performance of Workstation Cluster

3.3.3 Performance Analysis on IBM SP2

3.3.3.1 Architecture

We used the IBM SP2 parallel computer at NASA/Ames for this study. The NAS SP2 is essentially a group of IBM RS6000/590 workstations connected by a fast network with software that makes it appear like a single parallel computer. An SP2 computer can be composed of "wide" nodes which are RS6000/590 workstations or "thin" nodes which are RS6000/390 workstations. The NAS IBM SP2 consists entirely of wide nodes. Wide nodes have more expansion capability, but the main difference is that they have two or four times the memory bandwidth of thin nodes. The RS6000/590 workstation is based on a POWER2 multi-chip RISC processor. Features of the processor include:

- Clock rate: 66.7 MHz
- Data cache: 256 Kbytes (on wide nodes with 4 or 8 memory cards)
- Two integer computation units
- Two floating-point computation units

The nodes on an SP2 are connected by a high-performance switch. The switch can transfer data between SP2 nodes with 1 microsecond latency and 40 MB/s bi-directional bandwidth. Message passing is the only parallel programming paradigm currently supported by the IBM SP2.

3.3.3.2 Parallelization Environment/Tool

The same parallelization tool (xHPF/FORGE 90) available for workstation cluster is also

available on the NAS IBM SP2. As we had already parallelized both the SSM/I Precipitation Rate algorithm and the SeaWinds algorithm using xHPF on a distributed workstation cluster, it was a trivial exercise to port the parallelized SeaWinds code to the SP2.

3.3.3.3 Timing

3.3.3.3.1 I/O On Multiple Nodes

We tested two versions of SeaWinds, the first of which performed I/O operations from each node and did not distribute the data; the second version did I/O only from node 0 and distributed the data to the appropriate nodes. When I/O was performed on multiple nodes, the program scaled only to four processors. We believe that I/O overhead offset any gains made from parallel processing. Each node had to read the same input file, and the entire input file, before proceeding to the parallelized sections. The more processors we added, the worse the I/O times became as more and more processors tried to read the input file. Concurrent I/O is not recommended if all the processors have to read the same file, especially when the file is very large. Performance was very good when each processor read from a copy of the input file. However, with large data files many copies becomes prohibitively expensive storage-wise.

3.3.3.3.2 I/O On One Node

The execution times illustrated in Figure 3.3-7 are represented as serial + I/O (this includes data distribution across multiple CPUs from node 0), the time within the parallel sections of the program and system overhead includes time spent waiting for resource allocation. The system overhead includes any time spent waiting for system resources prior to program execution. The data distribution is included in the parallel sections because it is performed by directives inserted by the parallelization tool FORGE 90/xHPF. It can be seen that overhead time can be substantial when multiple processors are requested. It is not clear why the overhead is substantially higher for two processors. The run was verified by repeating several times on different days. We speculate that it may be due to the system configuration. The parallel sections, however, scale very nicely up to 64 processors. Communication speed can impact overall throughput if node 0 performs I/O and distributes data to all other nodes. We found that the IBM SP2 data distribution using PVM was slow and can seriously affect overall performance. However, with Message Passing Interface (MPI) and IBM's native Message Passing Library (MPL), our experience was more positive (discussed later under Purdue Benchmarks). MPL is IBM proprietary and is optimized for the SP2 architecture. Therefore, superior performance is expected. We could not repeat the SeaWinds run using MPI because parallelization was performed using FORGE 90/xHPF, which at the time of this work did not support MPI as a message passing library. Note that I/O, data distribution and other overhead are dependent on the system configuration.

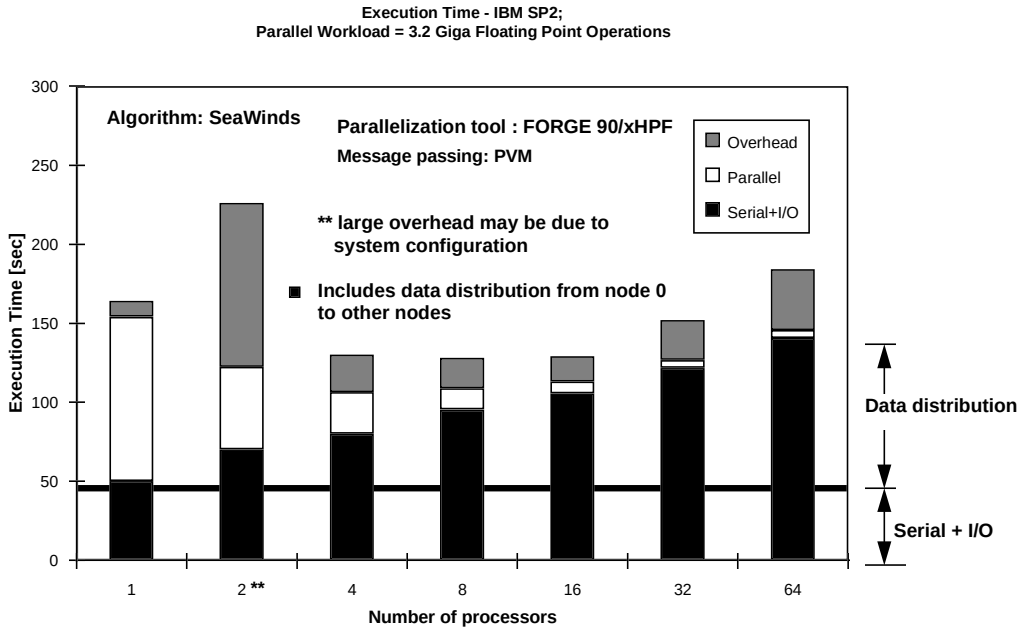


Figure 3.3-7. SeaWinds Execution Time on IBM SP2

3.3.3.4 Speedup

Figure 3.3-8 illustrates the speedup of the parallel sections and overall program on the IBM SP2. The ideal speedup is shown for reference. The speedup of the parallel sections of the program is very impressive on the IBM SP2, while overall speedup is discouraging. The degradation in overall performance is predominantly attributed to the poor performance in distributing data from node 0 to the various processors and the large overhead time associated during each execution. We found that PVM message passing library can degrade performance on the IBM SP2. Instead, MPI is recommended as a message passing library.

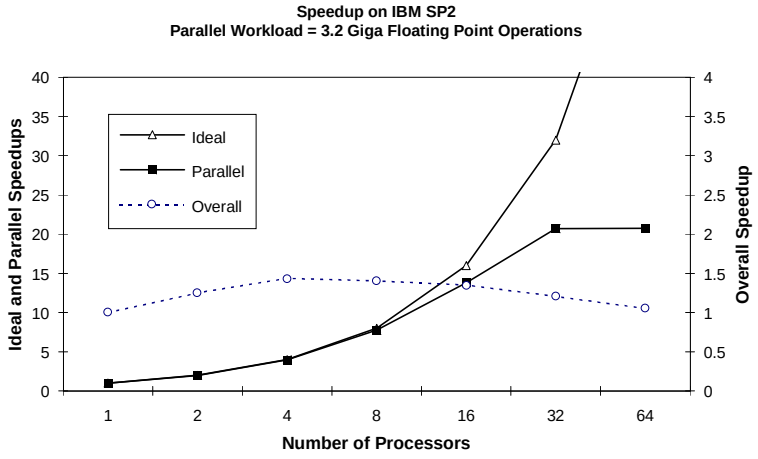


Figure 3.3-8. SeaWinds Speedup on IBM SP2

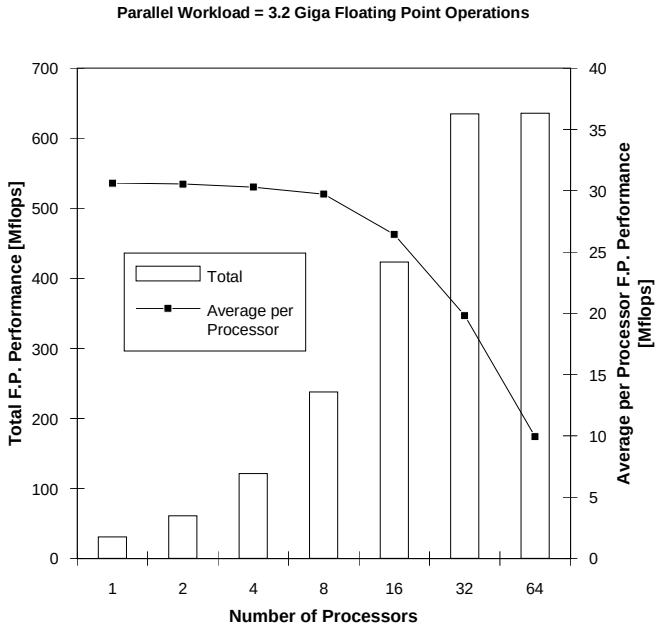


Figure 3.3-9. Floating Point Performance on IBM SP2 for SeaWinds

3.3.3.5 Floating Point Performance

The IBM SP2 has a vendor rated theoretical peak processor speed of 266 MFLOPS per processor. Figure 3.3-9 illustrates the total floating point performance (left scale) and the average per processor floating point performance (right scale) for the parallel sections of the SeaWinds program containing 3.2 Giga Floating Point Operations. The total floating point operations varied from 30 MFLOPS for a single processor to 640 MFLOPS for 64 processors. This translates to ~30 MFLOPS per processor when up to 8 processors are used. When 64 processors were used, floating point performance degrades to 10 MFLOPS per processor. The IBM SP2 operated at 11% of rated peak MFLOPS on 1 processor, decreasing to 4% for 64 processors.

3.3.4 Cray T3D

3.3.4.1 Architecture

We used the Cray T3D at JPL for this study. It is a 256-node system with a DECchip 21064 cache-based reduced instruction set computing (RISC) microprocessor rated at 150 MFLOPS peak performance per processor, with pipelined functional units that can issue multiple instructions per cycle, and supports 64-bit floating point arithmetic. Each Processing Element (PE) comprises of a DEC Alpha microprocessor with local memory. The PEs are connected by a

very fast bi-directional 3-D torus system interconnect network with peak interprocessor communication rates of 300 MB/s in every direction through the torus resulting in up to 76.8 GB/s of bisection bandwidth. Other features include:

- Multiple Instruction, Multiple Data (MIMD) architecture with Single Instruction Multiple Data (SIMD) support
- Physically distributed, globally addressable memory
- 2 to 4 I/O gateways with 1.6 GB/s peak I/O bandwidth
- Cray Research Adaptive Fortran (CRAFT) programming model with Data Parallel Programming Method and Message Passing with Cray optimized PVM
- Cray Tools with advanced programming utilities
- High-level performance analyzer

3.3.4.2 Parallelization Environment/Tool

The CRAFT programming model incorporates both the traditional data parallel and message passing MPP programming styles with the newer work sharing capability. Parallel work can thus be distributed at all levels within the program, from the subroutine to the individual loop, to an array. Parallelism can be explicitly expressed through message passing library calls, or implicitly through data parallel constructs and directives, taking advantage of the compiler to distribute data and work. For parallelization of SeaWinds, CRAFT was used with data parallel and work sharing capability, and through parallel constructs and directives, allowing the compiler to automatically distribute data and work.

There are other parallelization tools available for the Cray T3D. However, since FORGE 90/xHPF was not available at JPL at the time of this prototyping, we decided to use CRAFT instead to parallelize SeaWinds.

3.3.4.3 Timing

At JPL, the Cray YMP was the front-end machine for the Cray T3D. The Cray YMP received all jobs submitted to the Cray T3D and performed all I/O operations. The data were then sent to the Cray T3D for processing. Figure 3.3-10 illustrates the execution time for SeaWinds on the Cray T3D. The execution time is represented as the time taken for the serial portions including I/O, the parallel sections of the code and system overhead. The overhead includes wait time for resources before program execution. This includes wait time to get resources to begin program execution. The data distribution from processing element 0 is included in the parallel sections of the code. CRAFT inserts directives into the program to distribute data to the appropriate processing elements. The parallel sections of the code scale nicely and gives the best performance on 64 PEs. However, the time spent by the serial sections of the code including I/O is constant taking up approximately 27% of the total execution time on 1 processor. Sixty-four processors appears to be optimal for the given workload.

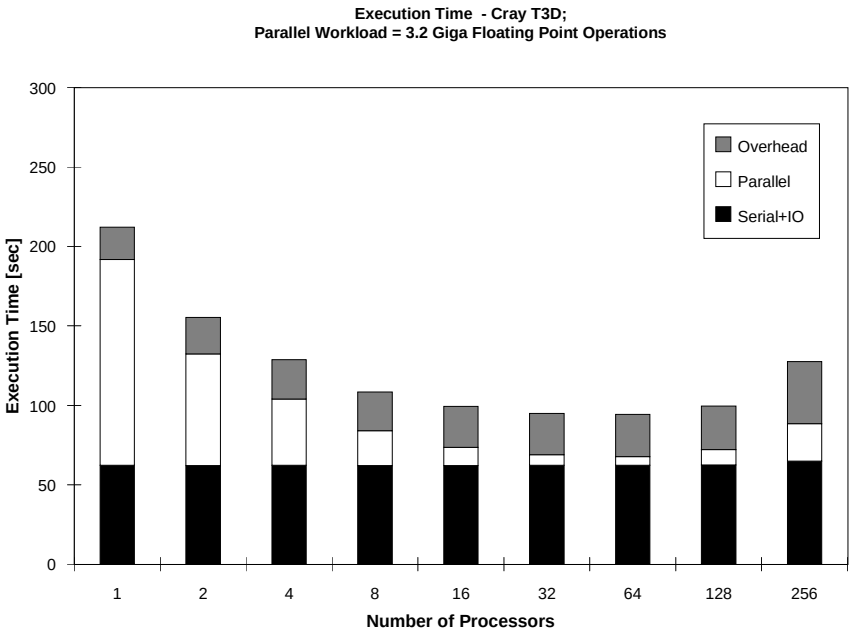


Figure 3.3-10. SeaWinds Execution Time on the Cray T3D

3.3.4.4 Speedup

Recall that speedup is defined as the ratio of the total execution time on *n* processors to that on a single processor. Figure 3.3-11 represents speedup as a function of number of processors for the parallel sections and for the entire program. The ideal speedup serves as reference. The speedup within the parallel sections of the program peaks to 25 for 64 processors. However, due to the large system overhead and the time spent performing I/O operations on the front-end Cray YMP, the overall speedup drops to 2.5 for 64 processors. Excluding configuration dependent parameters like system overhead and I/O operations on the front-end Cray YMP, the Cray T3D appears to be an impressive parallel computer.

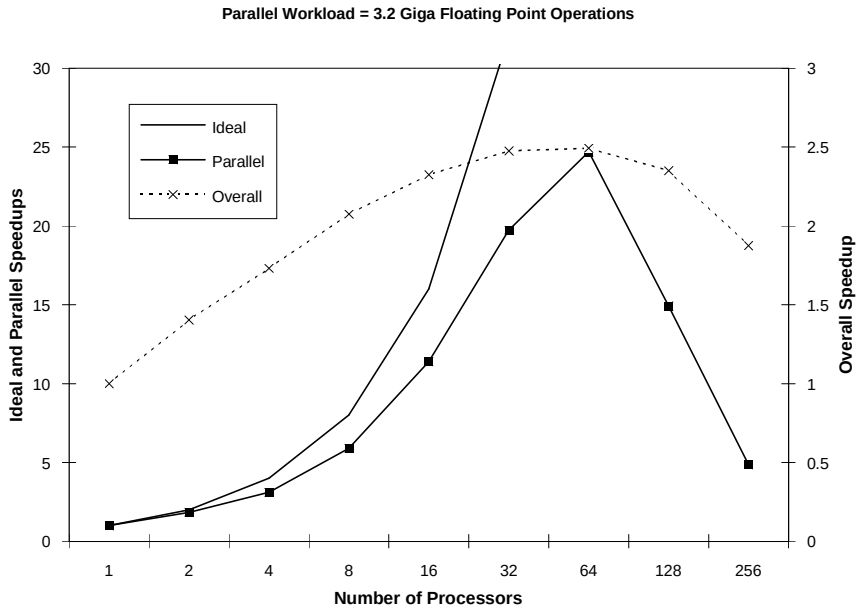


Figure 3.3-11. SeaWinds Speedup on the Cray T3D

3.3.4.5 Floating Point Performance

The peak floating point performance as rated by the vendor for each processor on the Cray T3D is 150 MFLOPS. Figure 3.3-12a shows the total floating point performance (left scale) and the average floating point performance per processor (right scale). The floating point performance is calculated within the parallel loop for a workload of 3.2 Giga Floating Point Operations. Floating point operations are used in a general sense and includes integer operations. The total floating point performance gradually increases from ~25 MFLOPS for one processor, peaks at 64 processors to ~600 MFLOPS and then falls rapidly for 256 processors. Interprocessor communication causes overall degradation in floating point performance. The floating point performance per processor decreases from ~17% of the rated peak MFLOPS for one processor to 7% for 64 processors.

We performed experiments to determine the effect of workload on floating point performance. Figures 3.3-12b-c illustrate the results. The parallel workload within the program was increased from 3.2 Giga Floating Point Operations to 407 Giga Floating Point Operations (see Figure 3.3-12b) and ultimately 3261 Giga floating point operations (see Figure 3.3-12c). The floating point performance per processor shows a dramatic improvement and asymptotically approaches 17 MFLOPs (11% of rated peak MFLOPS) even for large number of processors (up to 256). We found that the Cray T3D requires a large workload for multiprocessing (especially with large number of processors) in order to maintain the same performance as that for a single processor execution.

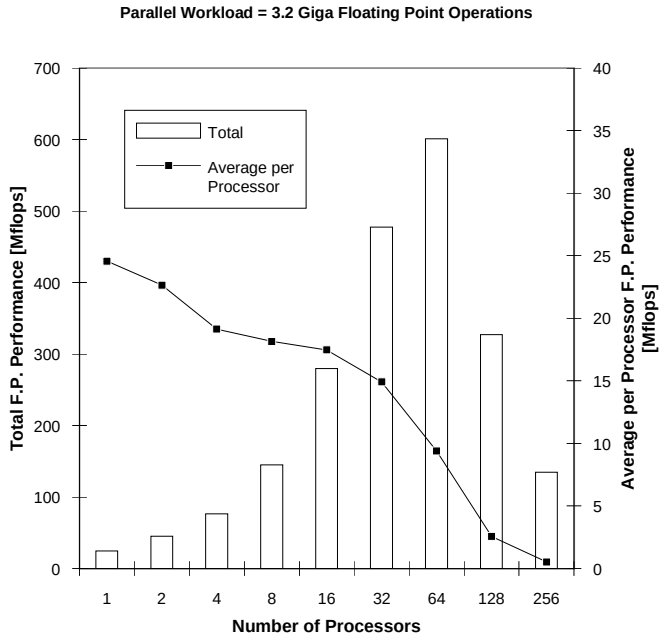


Figure 3.3-12a. Floating Point Performance of the Cray T3D for 3.2 Giga Floating Point Operations Workload

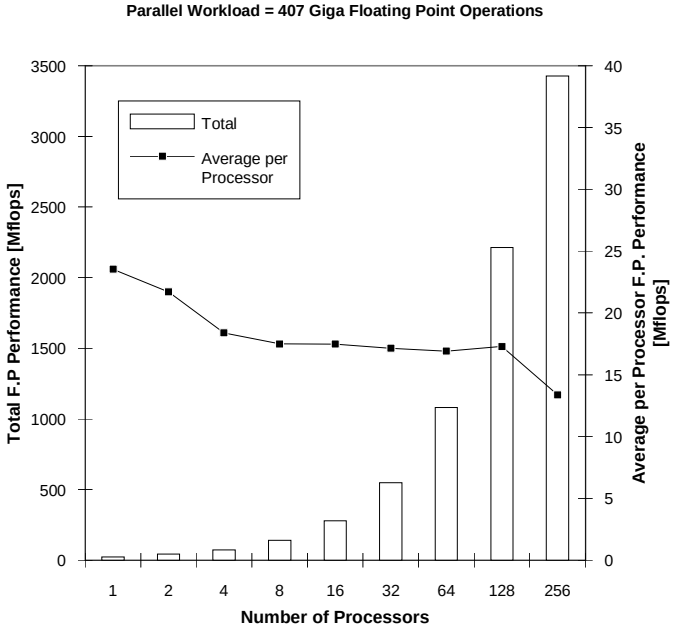


Figure 3.3-12b. Floating Point Performance of the Cray T3D for 407 Giga Floating Point Operations Workload

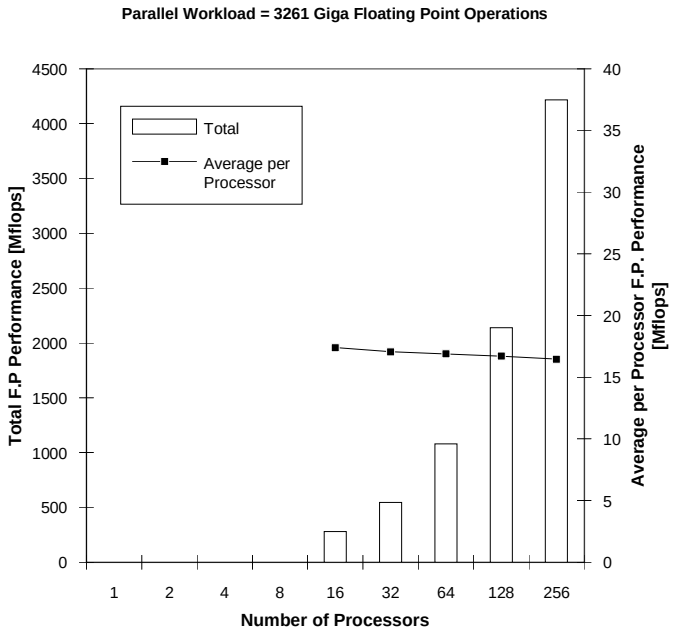


Figure 3.3-12c. Floating Point Performance of the Cray T3D for 3261 Giga Floating Point Operations Workload

3.3.2 Performance Comparison of Various Hardware Platforms

The performance of SGI Challenge XL, IBM SP2 and workstation cluster relative to the Cray T3D as a function of number of processors is shown in Figure 3.3-13. The performances of the entire program (solid lines) and the parallel sections (dashed lines) of the code are illustrated. The overall throughput performance of the SGI Challenge XL increases with increase in number of processors. The shared memory architecture of SGI Challenge makes interprocessor communication implicit. For 8 processors, performance is approximately 1.8 times greater than the Cray T3D. The Cray YMP, as the front-end to the Cray T3D performed all I/O operations. The overall throughput for workstation cluster (SGI Indigo, 2 HP735s and HP715) with concurrent I/O (each node reads input data independently and there is no large scale distribution of data from node 0) increases to about 1.2. This improves the overall performance relative to the Cray T3D. The workstation cluster with I/O on only one node (node 0), then distributing data to the other nodes demonstrates a dramatic degradation in performance. The IBM SP2 performed consistently slower than the Cray T3D for multiprocessing because data distribution is not as efficient as the Cray T3D when PVM was used as the message passing library. Further experiments with other benchmarking programs (Purdue benchmarks discussed later) indicates

that MPI and MPL can provide better performance on the IBM SP2. We could not test the use of MPI/MPL for SeaWinds because the version of Forge 90/xHPF we used did not support the use of MPI/MPL on the IBM SP2.

As a parallel processor (represented as dotted lines in Figure 3.3-13 for the parallel sections of the program), the Cray T3D and IBM SP2 are far more superior to the SGI Challenge. However, for small number of processors (less than 32), the IBM SP2 performs better than the Cray T3D. The performance is certainly better on the Cray T3D when large number of processors is used. Again, the reader should recall that PVM was used as a message passing library on the IBM SP2. Both MPI and MPL are proven to be better than PVM for message passing.

Figure 3.3-14 shows I/O and data distribution for the SGI Challenge XL, workstation cluster and IBM SP2 relative to the Cray T3D (I/O is performed by the front-end Cray YMP). The performance of I/O (1 processor) for the SGI Challenge XL is just over 3 times that of the Cray YMP. However, data distribution (for multiple processors) on the SGI Challenge is implicit because of the shared memory architecture. The SGI Challenge XL outperforms the Cray YMP. Workstation cluster with concurrent I/O (each node reads data and owns data to be processed) also performs better than the Cray. The IBM SP2 I/O (1 processor) is comparable to the Cray, but the poor data distribution speed (when PVM was used) decreases performance as the number of processors increased. Workstation clusters with serial I/O (data read on node 0 and distributed to other nodes by PVM) performs poorly due to excessive overhead associated with distributing data across an external interconnect. It is important to note that the system configuration can impact overall performance of a program.

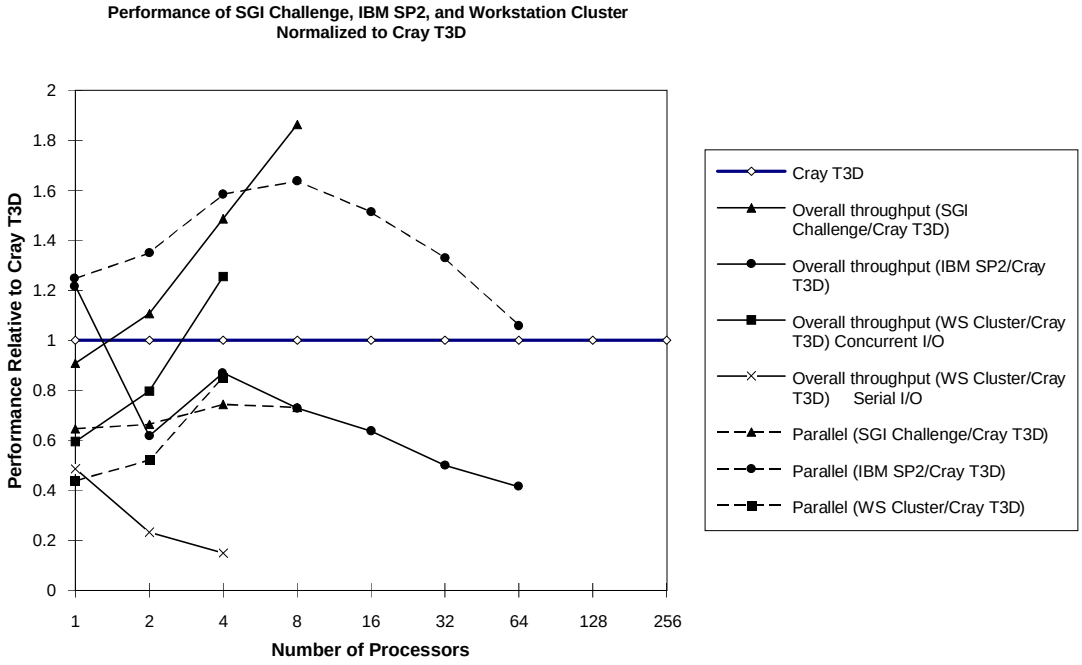
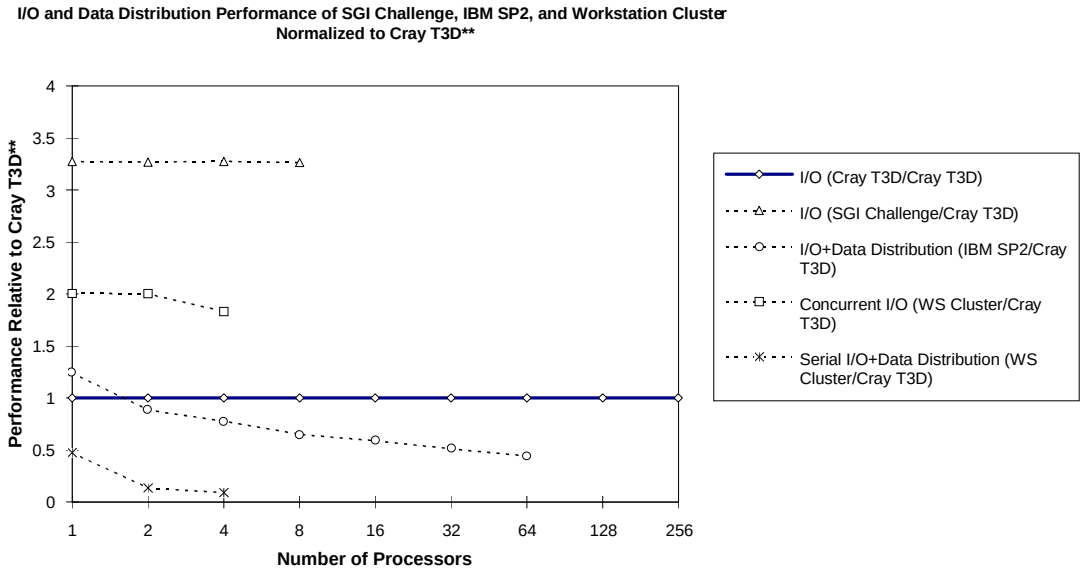


Figure 3.3-13. Processing Performance Comparison of Various Hardware Platforms



**** I/O for Cray T3D is performed by front-end Cray YMP**

Figure 3.3-14. I/O and Data Distribution Performance of Various Hardware Platforms

4. Caveat

- The SGI Challenge runs were made from a host attached disk
- The Cray T3D and IBM SP2 were not dedicated machines, and performance could be configuration dependent
- The Power Fortran Accelerator (PFA) was used as the parallelization tool on the SGI Challenge XL (FORGE 90/xHPF is available for the SGI Challenge XL but not at the ECS STL), CRAFT was used for parallelization on the Cray T3D (FORGE 90/xHPF is available for the Cray T3D but not at the JPL facility). A straight comparison of parallel code on all platforms with the same parallelization tool could not be achieved.
- CRAFT used a data parallel model with directives to perform parallelization on the Cray T3D. We did not use the message passing paradigm. However, on the IBM SP2, we used the PVM message passing library.

5. Purdue Benchmarks

5.1 Background

The Purdue benchmarks are a collection of a set of computational problems, that are simple, yet diverse, and selectively address different aspects of parallel computing, enabling systematic testing of the compiler. The benchmarking suite of fifteen problems was proposed by the Purdue University Group [5]. Most of the computational problems in this set have been extracted from larger computations, and may be artificial by themselves. Nevertheless, they represent a sampling of practical computations, even though not always the most efficient algorithm has been selected. In particular, they represent various schemes of data dependencies, from very simple application with almost no interprocessor communication required to more complex with large communication overhead, to irregular problems difficult to parallelize.

5.2 Purpose

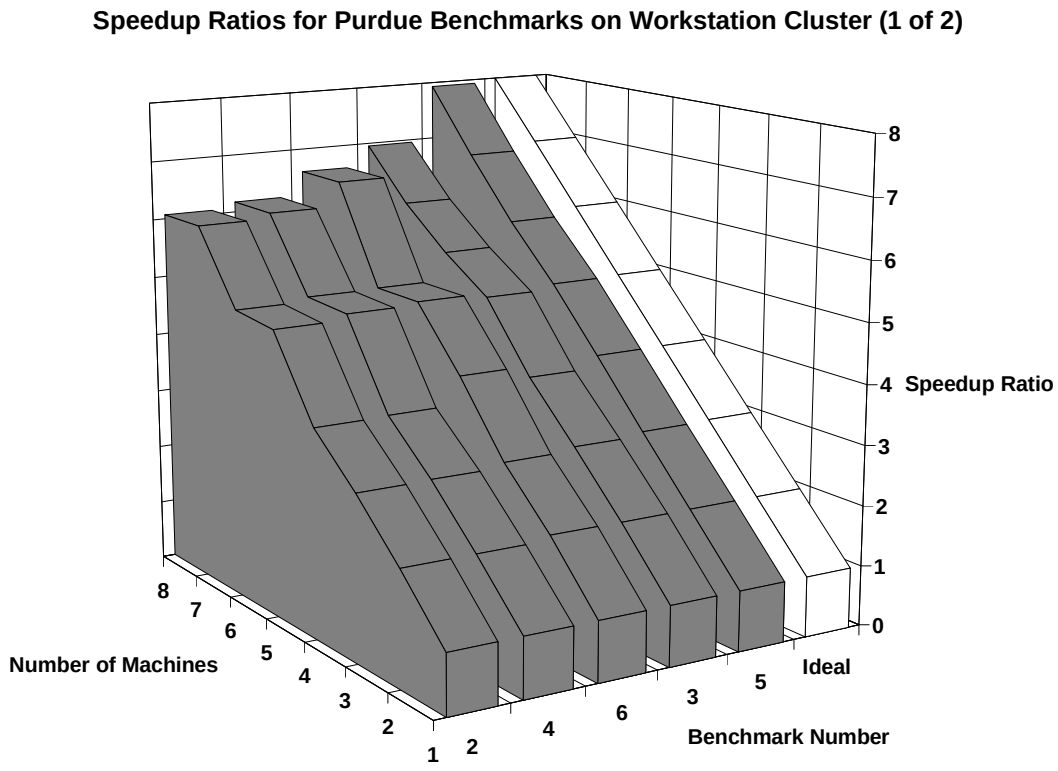
These benchmarking suite of algorithms are tested for automatic parallelization. Our goal is to demonstrate how the code should be structured for good performance, what types of code structures to avoid in a (data) parallel programming, and to evaluate automatic parallelization compilers that can simplify programming for parallel computers. They are useful for training purposes as well as to test new parallelization compilers. We have used the Purdue set Fortran 77 benchmarks. Benchmarks in Fortran 90, Fortran 90D and Fortran for Massively Parallel Processors using message passing are also available. Refer to [3] for description of the Purdue Benchmarks.

5.3 SGI Challenge

Refer to [3] for Purdue Benchmark runs on the SGI Challenge XL.

5.4 Workstation Cluster

The automatic parallelization option available on Forge 90/xHPF was used to parallelize the Purdue benchmarks. Figure 5.4-1 shows performance of the benchmarks in a distributed memory workstation cluster environment. PVM was used for message passing. Table 3.3-1 lists the workstations in the cluster.



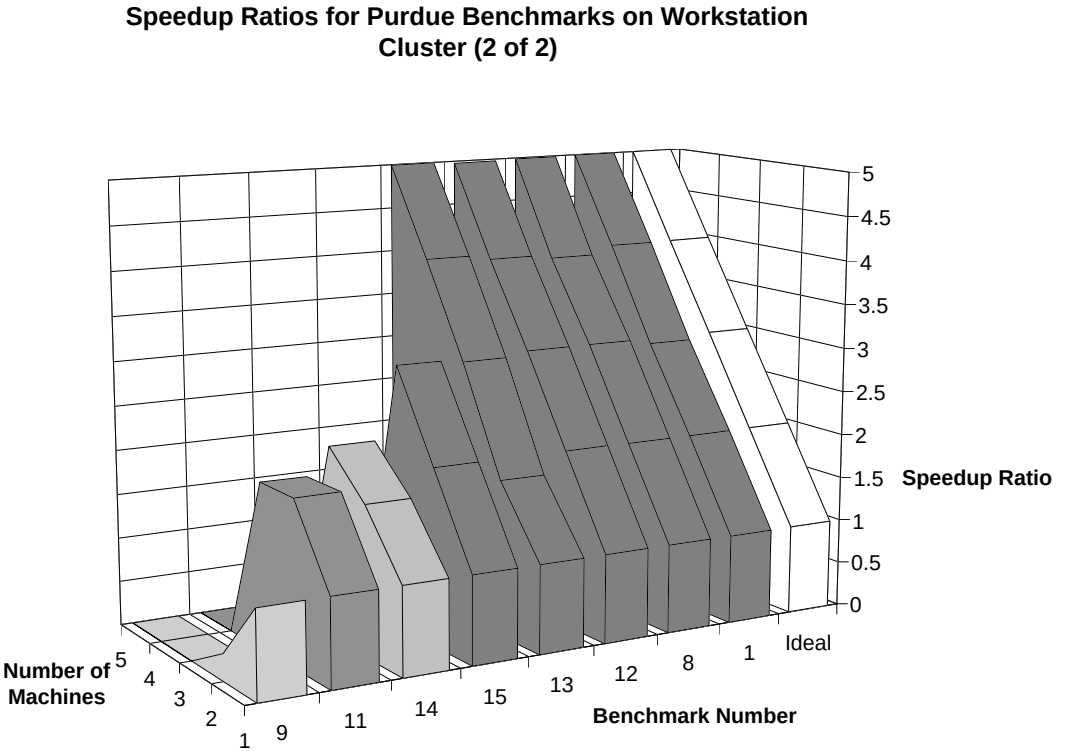


Figure 5.4-1. Speedup of Purdue Benchmarks on Workstation Cluster

5.5 Cray T3D

Figure 5.5-1 illustrates the behavior of selected Purdue Benchmark problems on the Cray T3D. CRAFT was used to automatically parallelize the problems.

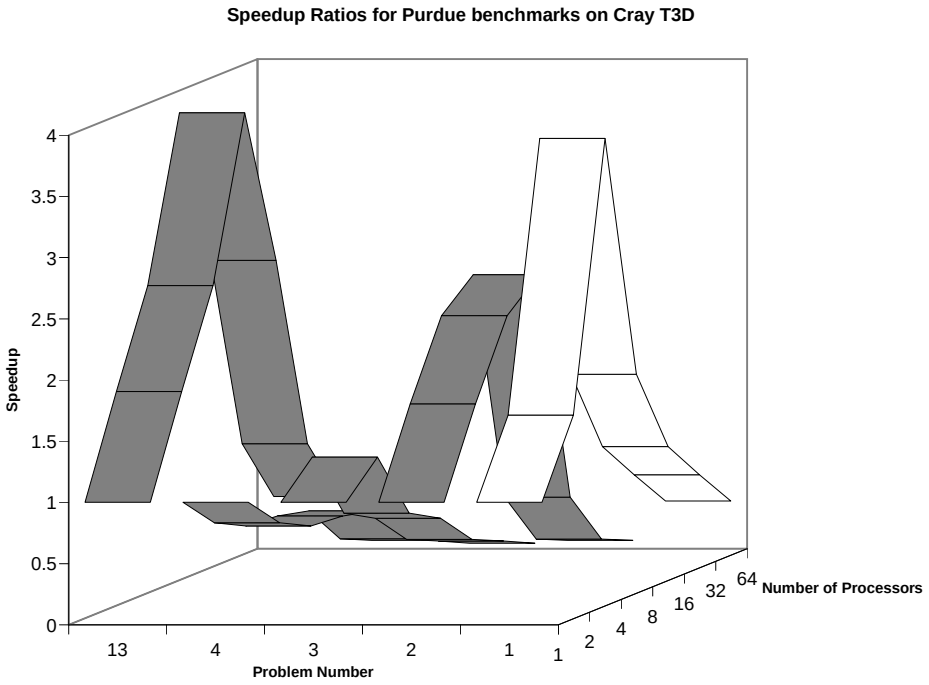


Figure 5.5-1 Speedup of Purdue Benchmarks on the Cray T3D

5.6 IBM SP2

Unlike the Cray T3D which allows for shared memory processing (called distributed shared memory) within its distributed memory architecture, the IBM SP2 is purely a distributed memory machine. This means that explicit message passing libraries should be used to allow for communication among the processors within parallel sections of the code. There are a number of message passing libraries available today. PVM has been the most widely available and used library for message passing. More recently, MPI has become more popular because of its standardized interfaces. There are also commercial packages like Linda and Express for message passing. One of the factors that can affect the performance of a program is the message passing library used. Results from benchmarking work done at NAS at NASA/Ames have indicated that MPL, a native message passing library available on the IBM SP2 is far superior to any other message passing library. Therefore, we decided to use MPL for measuring performance of the Purdue Benchmarks, although using MPL restricts portability of the code (xHPF/Forge 90 was not used to parallelize the Purdue benchmarks on the IBM SP2 because it currently does not support MPL). The speedup due to parallelization using MPL is illustrated in Figure 5.6-1.

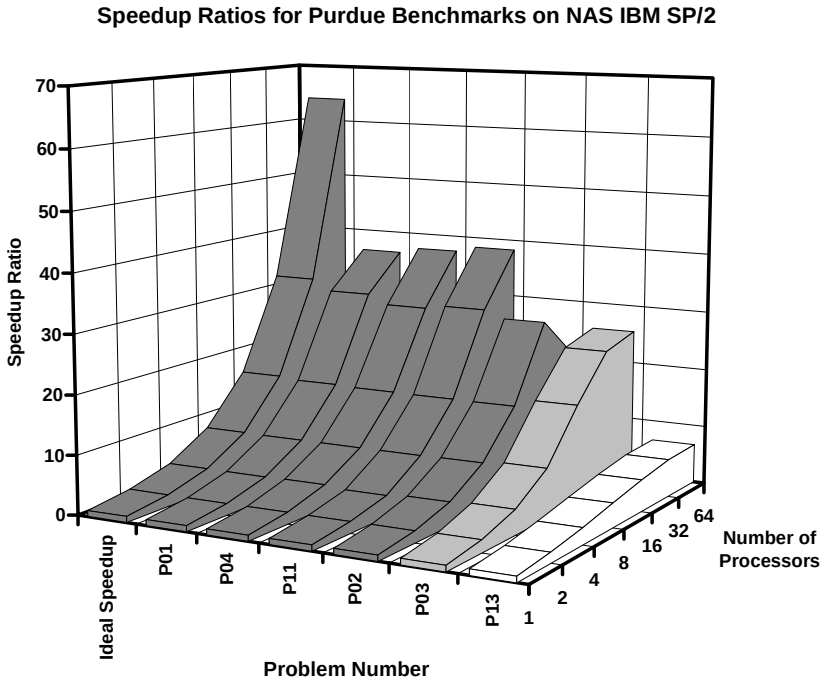


Figure 5.6-1 Speedup of Purdue Benchmarks on the IBM SP2

6. Summary and Conclusions

A heritage ECS science software from JPL (SeaWinds) was analyzed for suitability to parallel processing. A parallelization strategy was developed and implemented on the SMP, workstation cluster, Cray T3D and IBM SP2 MPPs using native and third-party parallelization tools. The performance of the program was studied under various processing alternatives. Using SeaWinds as a benchmark, the actual MFLOPs delivered by the various hardware platforms was compared with the rated theoretical peak MFLOPs. It was found the SeaWinds Level 2 science software is highly scalable for multiple processors and is suitable for processing in parallel. The conclusions are listed below:

- Parallelization developed for SMP/workstation cluster is applicable for MPPs
- Migration to MPPs is simplified if the same tool that were used to develop parallel code on SMP and workstation cluster is available on the MPPs
- MPP floating point performance in multiprocessing mode is affected by processing workload. For smaller workloads, floating point performance degrades with increase in number of processors.
- The SGI Challenge XL was better than the Cray T3D and IBM SP2 (although they are dependent on the system configuration) in terms of overall throughput. But the Cray T3D and IBM SP2 are certainly better parallel machines.
- Workstation cluster with concurrent I/O (each node reads input data and, therefore, owns the data it processes) appears promising and has the potential to equal MPPs in performance
- Front-end machines and disk configuration can impact I/O performance and introduce substantial overhead in MPPs
- The IBM SP2 using PVM message passing is slow with data distribution across multiple processors. For SeaWinds, the IBM SP2 performed better than the Cray T3D for small number of processors. However, performance degraded for large number of processors.

7. References

[1] PDPS Prototyping at the ECS Science and Technology Laboratory: Progress Report #2, April 1994.

[2] PDPS Prototyping at the ECS Science and Technology Laboratory: Progress Report #3, June 1994.

[3] PDPS Prototyping at the ECS Science and Technology Laboratory: Progress Report #4, September 1994, #194-00569TPW.

[4] Power Fortran Accelerator™ User's Guide, Silicon Graphics, Inc.

[5] Rice, J. R., and Jing, J. Problems To Test Parallel and Vector Languages. Technical Report., CSD-TR-1016, 1990.

8. Acknowledgments

The Cray Supercomputer used in this investigation was provided by the Jet Propulsion Laboratory (JPL), Pasadena, CA under funding from the NASA offices of Mission to Planet Earth, Aeronautics and Space Science.

The IBM SP2 used in this investigation was provided by NASA/Ames, Moffet Field, CA under funding from the NASA offices of Mission to Planet Earth and Space Science.

Drs. Steve Gunter and Scott Dunbar of JPL are acknowledged for providing SeaWinds software for ECS prototyping.

Marek Chmielowski and Scott Bramhall of the ECS team are sincerely appreciated for their diligence in programming and implementing SeaWinds in multiprocessing mode on a variety of platforms.

Abbreviations and Acronyms

ADEOS	Advanced Earth Observing System
AMSR	Advanced Microwave Scanning Radiometer
APR	Applied Parallel Research
CPU	Central Processing Unit
DAAC	Distributed Active Archive Center
DCE	Distributed Computing Environment
DMP	Distributed Memory Parallelizer
DWM	Digital Weather Model
ECS	EOSDIS Core System
EOSDIS	Earth Observing System Data Information System
ESDIS	Earth Science Data and Information System
FDDI	Fiber Distributed Data Interface
I/O	Input/Output
IBM	International Business Machines
Ir1	Interim Release-1
IT	Instrument Team
JPL	Jet Propulsion Laboratory
KB	Kilo Bytes
MB	Mega Bytes
MIMD	Multiple Instruction, Multiple Data
MPI	Message Passing Interface
MPL	Message Passing Library
MPP	Massively Parallel Processor
NAS	Numerical Aerodynamic Simulation
NASA	National Aeronautics and Space Administration
NSCAT	NASA Scatterometer
OSF	Open Software Foundation
PDPS	Planning and Data Processing System

PFA	Power Fortran Accelerator
PVM	Parallel Virtual Machine
RISC	Reduced Instruction Set Computer
SGI	Silicon Graphics, Inc.
SIMD	Single Instruction, Multiple Data
SMP	Symmetric Multiprocessor
SPMD	Single Program, Multiple Data
SSM/I	Special Sensor Microwave/Imager
STL	Science and Technology Laboratory